Mixed-Integer Linear Programming Model for the Super-Colored Path Problem in Digraphs

Rafael Marian, Wesly Carmesini Ataide, Rafael de Santiago, Álvaro Junio Pereira Franco

[1]Depto. de Informática e Estatística Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-370 – Florianópolis – SC – Brazil

rafael.marian@grad.ufsc.br, weslycarmesiniataide@gmail.com

{r.santiago,alvaro.junio}@ufsc.br

Abstract. *This paper presents a method based on Mixed-Integer Linear Programming (MILP) to solve the Super-Colored Path Problem. Given a digraph D with colored vertices, and for each vertex v in D, we want to find the maximum number of distinct colors that a simple path starting from v can take. The proposed method draws resemblance with the classic formulation for solving the Traveling Salesman Problem, in particular the presence of subtours.*

1. Introduction

There are a variety of applications for vertex-colored graph problems. One such problem is the Super-Colored Path Problem (SCPP) [Franco and Vendramin 2021] which is defined as follows. *Given a vertex-colored digraph D, determine for each vertex v the maximum number of colors on a directed path starting at v.* This problem is NP-hard and was defined to support the development of a parallel algorithm for finding chromatic rings in kinship networks [Vendramin 2021]. In this setting, some properties of individuals are represented by colors and chromatic rings can be considered in the analysis [Franco and Vendramin 2022].

Integer or linear programming is commonly used to solve problems in graph theory, such as graph layout mapping [Gurski 2015] and map coloring with varied prerequisites [Yao 2023]. A method was previously proposed to solve the SCPP restricted to DAGs, based on an integer programming model that computes the value for a given vertex [Ataide et al. 2024]. Here, we extend the model for general digraphs, drawing on the literature related to the *Traveling Salesman Problem* (TSP), with a key similarity being the occurrence of *subtours*, paths that are not connected to the solution path but may be incorrectly counted as part of the solution. This relationship between the proposed model and TSP formulations allows the use of existing techniques for handling *Subtour Elimination Constraints* (SECs). In particular, the sets of SECs known as *Miller–Tucker–Zemlin* (MTZ) [Miller et al. 1960] and *Dantzig–Fulkerson–Johnson* (DFJ) [Dantzig et al. 1954] have been applied.

Now we describe the problem addressed in this work, which is slightly different from the SCPP. Given a vertex-colored digraph $G = (V, A, K)$, where V is the set of vertices, A is the set of arcs, and K is the set of colors, we define $super(v)$ for $v \in V$ as the maximum number of distinct colors that can appear in a simple path starting at v. A path starting at v is called v-super-colored if it contains exactly $super(v)$ distinct colors.

In this work, we present a method for computing $super(v)$ for a given vertex $v \in V$. In contrast, the SCPP requires computing $super(v)$ for all vertices $v \in V$. Therefore, the SCPP can be solved by applying our method independently to each vertex $v \in V$.

2. Mixed-Integer Linear Programming Model

We begin by defining $G_r = (V_r, A_r, K_r)$ a subgraph of G induced by the vertices and arcs reachable from r, excluding arcs that return to r, i.e., arcs of the form (v, r) for $v \in V$. We then define $G'_r = (V'_r, A'_r, K_r)$ as a copy of G_r augmented with a sink vertex t and additional arcs, where $V'_r = V_r \cup \{t\}$ and $A'_r = A_r \cup \{(v, t) : v \in V_r \setminus \{r\}\}$. This sink vertex is used to enumerate simple paths starting at r.

Let $x_{v,u} \in \{0, 1\}$, for all $(v, u) \in A'_r$ be a set of binary variables such that $x_{v,u} = 1$ if and only if the arc (v, u) belongs to a simple selected path starting at r. The variables $y_k \in \{0, 1\}$, for all $k \in K_r$, describe the presence of the color k on the selected path. For a given set V of vertices, we denote by $V^k \subseteq V$ the set of vertices with color k. Furthermore, let $c_{v,u} \in \mathbb{R}^+$, for all $(v, u) \in A'_r$ be real variables that represent the order of arcs on the solution path. Finally, the model is defined as follows:

$$\text{maximize} \sum_{k \in K_r} y_k \tag{1}$$

$$\text{subject to} \sum_{(r,v) \in A'_r} x_{r,v} = 1 \tag{2a}$$

$$\sum_{(v,t) \in A'_r} x_{v,t} = 1 \tag{2b}$$

$$\sum_{(u,v) \in A'_r} x_{u,v} \leq 1, \qquad \forall v \in V'_r \setminus \{r, t\} \tag{3a}$$

$$\sum_{(v,u) \in A'_r} x_{v,u} \leq 1, \qquad \forall v \in V'_r \setminus \{r, t\} \tag{3b}$$

$$\sum_{(u,v) \in A'_r} x_{u,v} = \sum_{(v,u) \in A'_r} x_{v,u} \qquad \forall v \in V'_r \setminus \{r, t\} \tag{4}$$

$$\sum_{v \in V^k_r} \sum_{(v,u) \in A'_r} x_{v,u} \geq y_k \qquad \forall k \in K_r \tag{5a}$$

$$\sum_{(v,u) \in A'_r} x_{v,u} \leq y_k \qquad \forall k \in K_r, \forall v \in V^k_r \tag{5b}$$

$$c_{v,u} \leq |V'_r| x_{v,u} \qquad \forall (v, u) \in A'_r \tag{6a}$$

$$\sum_{(v,u) \in A'_r} c_{v,u} \geq \sum_{(u,v) \in A'_r} c_{u,v} + \sum_{(v,u) \in A'_r} x_{v,u} \qquad \forall v \in V_r \tag{6b}$$

$$c_{r,v} = x_{r,v} \qquad \forall (r, v) \in A'_r \tag{6c}$$

Alternatively, (6a), (6b), and (6c) can be replaced with a single constraint that reflects a DFJ formulation in an Integer Program [Dantzig et al. 1954]. Similarly to the TSP, these restrictions scale in number exponentially, but can be added using branch-and-cut (lazy constraints) solver implementations, which makes them applicable. Take the set C to be the set of all cycles within G_r:

$$\sum_{(v,u)\in P} x_{v,u} \leq |P| - 1 \qquad \forall P \in C \tag{7}$$

2.1. Discussions on Model Correctness

The set (6a)-(6c) eliminate subtours from the solution space using a technique similar to the one found in traditional MTZ formulations for the TSP problem, with variables c that can be real or non-negative integers as in [Miller et al. 1960]. The constraint 6a establishes an upper limit to the order variables, while also correlating order variables with the path selection variables, since if $x_{a,b} = 0$ the right-hand side is 0. (6b) ensures an ever-incrementing order to the variables in a given path, and (6c) creates a starting value for the order variables.

Theorem 1. *The constraints (6a)-(6c) eliminate subtours from sets x and c of variables that follow them.*

Alternative to the above, (7) eliminates subtours on a case-by-case basis. Subtours only occur when there are cycles in our graph, thus by restricting the selection of variables to explicitly not include an entire cycle, for all the cycles in our graphs, eliminates subtours.

Theorem 2. *Constraint (7) eliminates subtours from a set of binary variables x.*

Constraints (2a)-(3b) relate to path selection, with (4) acting as a flow conservation constraint, while (5a) and (5b) to the colors present in the selected path. Thus, with the elimination of subtours from our solution space using Theorem 1 or 2, we can use these constraints to enumerate all simple paths outgoing from r.

Theorem 3. *Given any set of binary variables x conforming to (2a)-(4) in G_r, there is a corresponding simple path starting from r. Conversely, for any simple path starting from r, there is a set of variables x that conform to the same constraints and correspond to that path (for $|V| > 1$).*

The pair (5a) and (5b) enforce the selection of a set of y binary variables only to colors present in the path, such that $super(r) = \sum_{k\in K_r} y_k$.

Theorem 4. *A set of binary variables x that conform to (2a)-(3b) and binary variables y that conform to (5a) and (5b) will select colors if and only if there are vertices that are part of the selected path.*

Given all this, taking variable sets x, y, and c that follow (2a)-(6c) (or alternatively (7)) in G'_r, and using the objective function 1, the path formed by the edges (a, b) in which $x_{a,b} = 1$, except for (a, t), is a r-super-colored path and the value of the objective function is $super(r)$.

3. Final Remarks

The models were tested in comparison with a baseline algorithm for general digraphs found in [Ataide et al. 2024]. Due to the lack of real use-case instances with cycles, the

set was randomly generated using the `NetworkX 3.2.1` Python 3 package, using the "binomial graph" algorithm with the parameter $p = 0.30$, creating graphs with an average density of 30%. The graphs were then randomly colored with $|K| = \min(\lceil \frac{|V|}{4} \rceil, 10)$.

The algorithms were implemented in the Python 3 programming language, running `CPython 3.13.5`. For the linear programs, the solver of choice was the academic version of Gurobi Optimizer 12.0.3, using a machine with 8 gigabytes of RAM and an AMD Ryzen 5700U 8-core processor. To provide a better comparison between the linear programs, the pre-solving parameter was turned off. A set of 10 graphs was generated for each size, and the average of their execution time was taken. The code and graphs used are available online at `https://codigos.ufsc.br/rafael.marian/etc26`.

Table 1. Execution time for solving digraphs with the combinatorial algorithm in [Ataide et al. 2024], the MILP with the set (6a)-(6c), and with (7).

Vertex Count	Combinatorial	Model (DFJ)	Model (MTZ)
10	0.03s	0.07s	0.07s
15	5.68s	0.14s	0.13s
20	*	0.21s	0.30s
30	*	0.70s	1.98s
40	*	2.34s	5.88s
50	*	4.58s	11.74s
60	*	7.82s	26.38s
70	*	14.27s	50.72s
80	*	21.45s	89.94s

* The process was killed due to memory usage before being finished.

Similar to the corresponding TSP implementation of these models, there is a performance disparity between the DFJ and MTZ model. This is likely due to the higher amount of cuts added in by the DFJ model, drastically reducing the amount of simplex iterations required, thus reducing necessary computational time. Graph structures that show particular weaknesses to each model were not explored in this work.

In conclusion, proposed model is faster than a naive implementation of a combinatorial algorithm to solve this problem on generalized digraphs, and uses much less memory on a standard MILP solver. Future work should focus on finding further real-world applications of the problem, and revising existing MILP literature to apply known optimization techniques.

Acknowledgment

This work was partially supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico). The authors thank Rafael C. S. Schouery for noting that the model presented by Ataide could be generalized to general digraphs considering flow constraints.

References

Ataide, W. C., Santiago, R. d., and Franco, A. J. P. (2024). Algorithms for super-coloring in directed graphs. *Proceedings of LVI Brazilian Symposium on Operations Research*.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

Franco, Á. J. P. and Vendramin, M. E. (2021). Super-colored paths in digraphs. *Anais do VI Encontro de Teoria da Computação*, pages 94–97.

Franco, Á. J. P. and Vendramin, M. E. (2022). Uma experiência com redes de parentesco totalmente coloridas. In *REACT – VIII Reunião de Antropologia da Ciência e da Tecnologia*.

Gurski, F. (2015). Linear programming formulations for computing graph layout parameters. *The Computer Journal*, 58(11):2921–2927.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329.

Vendramin, M. E. (2021). Algoritmos para encontrar anéis cromáticos em redes de parentesco. Trabalho de Conclusão de Curso.

Yao, L. (2023). Optimizing map coloring: Using linear programming to find the minimum number of colors. *Theoretical and Natural Science*, 28:1–9.

A. Proofs

Theorem 1

Proof. Assume that there is a subtour $C = (v_0, \ldots, v_n, v_0)$ present in the variable set x. Considering the set of real variables c, take any value that follows (6a) and see from (6b):

$$\begin{aligned} c_{v_0,v_1} &\geq c_{v_n,v_0} + 1 \\ c_{v_1,v_2} \geq c_{v_0,v_1} + 1 &\implies c_{v_1,v_2} \geq c_{v_n,v_0} + 2 \ldots \\ \ldots c_{v_n,v_0} \geq c_{v_{n-1},v_n} + 1 &\implies c_{v_n,v_0} \geq c_{v_n,v_0} + n \end{aligned}$$

a contradiction. □

Theorem 2

Proof. Assume that there is a selected subtour $C = (v_0, \ldots, v_n, v_0)$. Then, $x_{v_0,v_1} = \cdots = x_{v_{n-1},v_n} = x_{v_n,v_0} = 1$. As such, we have for the application of (7) on C:

$$x_{v_0,v_1} + \cdots + x_{v_{n-1},v_n} + x_{v_n,v_0} \leq n + 1 - 1$$
$$n + 1 \leq n$$

a contradiction. □

Theorem 3

Proof. Take any v_o in that $(r, v_0) \in E_r$ and in which (2a) applies, so that we have $x_{r,v_0} = 1$. Then, with any neighbor v of v_0, we can assign $x_{v_0,v} = 1$ to satisfy (4). If $v = t$, we have constructed the path (r, v_0, t) that corresponds to the simple path (r, v_0). Otherwise, apply the previous step to v, and follow the same procedure until your path ends in t (which it must because of (2b)). Notice, also, that our chosen v must not be present in the path because of (3a), and that we cannot simultaneously choose 2 vertices v because of (3b). Then, you will find a path $(r, v_0, \ldots, v_n, t)$ corresponding to $(r, v_0, \ldots, v_n)$.

Conversely, choose a simple path $P = (r, v_0, \ldots, v_n)$, we can assign $x_{r,v_0} = 1$, $x_{v_n,t} = 1$ (satisfying (2a) and (2b)), and then for every $i \in \{0, \ldots, n-1\}, x_{v_i,v_{i+1}} = 1$ (satisfying (4)). See that, because P is simple, (3a) and (3b) are respected, since every vertex only appears exactly once in this path. □

Theorem 4

Proof. Assume that there is some color k for which $y_k = 1$, where there is no such $v \in V_r^k$ such that for any $u \in V$, $x_{v,u} = 1$. See, however, that the left-hand side of (5a) would equal to 0, and thus we would reach a contradiction. Conversely, suppose $y_k = 0$ but $x_{v,u} = 1$, for some $v \in V_r^k$. See now that in (5b), the left-hand side would equal 1, finding yet another contradiction. □